PATTERN SEARCH METHODS IN THE PRESENCE OF DEGENERACY *

MARK A. ABRAMSON †, OLGA A. BREZHNEVA ‡, AND J. E. DENNIS JR. §

Abstract. This paper deals with generalized pattern search (GPS) algorithms for linearly constrained optimization. At each iteration, the GPS algorithm generates a set of directions that conforms to the geometry of any nearby linear constraints. This set is then used to construct trial points to be evaluated during the iteration. In previous work, Lewis and Torczon developed a scheme for computing the conforming directions, but it assumed no degeneracy near the current iterate. The contribution of this paper is to provide a detailed algorithm for constructing the set of directions whether or not the constraints are degenerate. One difficulty in the degenerate case is in classifying constraints as redundant and nonredundant. We give a short survey of the main definitions and methods for treating redundancy and propose an approach to identify nonredundant ε-active constraints, which may be useful for other active set algorithms. We also introduce a new approach for handling nonredundant linearly dependent constraints, which maintains GPS convergence properties without significantly increasing computational cost. Some simple numerical tests illustrate the effectiveness of the algorithm. We conclude by briefly considering the extension of our ideas to nonlinear constraints with linearly dependent constraint gradients.

Key words. Pattern search, linearly constrained optimization, derivative-free optimization, degeneracy, redundancy, constraint classification

AMS subject classifications. 65K05, 49M30, 90C30, 90C56

1. Introduction.

This paper continues the development of generalized pattern search (GPS) algorithms [4, 16] for linearly constrained optimization problems

$$\min_{x\in\Omega} f(x)\,, \tag{1.1}$$

where $f:\mathbb{R}^n\to\mathbb{R}\cup\{\infty\}$ may be discontinuous, and the feasible region is given by

$$\Omega=\{x\in\mathbb{R}^n\,:\,a_i^Tx\le b_i,\ i\in I\}=\{x\in\mathbb{R}^n\,:\,A^Tx\le b\}, \tag{1.2}$$

where, for $i\in I=\{1,2,\ldots,|I|\}$, $a_i\in\mathbb{R}^n$, $b_i\in\mathbb{R}$, and $A\in\mathbb{Q}^{n\times|I|}$ is a rational matrix. In [4, 16], the feasible region Ω is defined as $\Omega=\{x\in\mathbb{R}^n\,:\,\ell\le\widehat{A}x\le u\}$, where $\widehat{A}\in\mathbb{Q}^{m\times n}$ is a rational matrix, $\ell,u\in\{\mathbb{R}\cup\{\pm\infty\}\}^m$, and $\ell<u$. As is evident, (1.2) reduces to the definition in [4, 16], where the rth row $\widehat{a}_r$ of the matrix $\widehat{A}$ is equal to some ith row a_i^T of the matrix A^T with a coefficient of $+1$ or -1, and $b_i=u_r$ or $b_i=-\ell_r$.

We target the case when the function $f(x)$ may be an expensive "black box", provide few correct digits, or may fail to return a value even for feasible points $x\in\Omega$. In this situation, the accurate approximation of derivatives is not likely to be practical. GPS algorithms rely on simple decrease in $f(x)$; *i.e.*, an iterate $x_{k+1}\in\Omega$ satisfying $f(x_{k+1})<f(x_k)$ is considered successful.

Lewis and Torczon [16] introduced and analyzed the generalized pattern search for linearly constrained minimization problems. They proved that if the objective function is continuously differentiable and if the

* Date: August 19, 2005

† Department of Mathematics and Statistics, Air Force Institute of Technology, AFIT/ENC, Building 641, 2950 Hobson Way, Wright-Patterson AFB, Ohio 45433 (mark.abramson@afit.edu, http://en.afit.edu/enc/Faculty/MAbramson/abramson.html)

‡ Department of Mathematics and Statistics, Miami University, 123 Bachelor Hall, Oxford, Ohio 45056, (brezhnoa@muohio.edu).

§ Computational and Applied Mathematics Department, Rice University - MS 134, 6100 Main Street, Houston, Texas, 77005-1892 (dennis@rice.edu, http://www.caam.rice.edu/~dennis). The research of this author was supported in part by AFOSR F49620-01-1-0013, the Boeing Company, Sandia CSRI, ExxonMobil, the LANL Computer Science (LACSI) contract 03891-99-23, by the Institute for Mathematics and its Applications with funds provided by the National Science Foundation, and by funds from the Ordway Endowment at the University of Minnesota.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2003		**00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Pattern Search Methods in the Presence of Degeneracy	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Air Force Institute of Technology,Department of Mathematics and Statistics,2950 Hobson Way Building 640,Wright Patterson AFB,OH,45433-7765**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**22**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

set of directions that defines a local search is chosen properly with respect to the geometry of the boundary of the feasible region, then GPS has at least one limit point that is a Karush-Kuhn-Tucker point. By applying the Clarke nonsmooth calculus [11], Audet and Dennis [4] simplified the analysis in [16] and introduced a new hierarchy of convergence results for problems with varying degrees of nonsmoothness. Second-order behavior of GPS is studied in [2].

Generalized pattern search algorithms generate a sequence of iterates $\{x_k\}$ in $\mathbb{R}^n$ with non-increasing objective function values. In linearly constrained optimization, a set of directions that defines the so-called POLL step must conform to the geometry of the boundary of the feasible region. The key idea, which was first suggested by May in [18] and applied to the GPS in [16], is to use as search directions the generators of cones polar to those generated by the normals of faces near the current iterate.

Lewis and Torczon [16] presented an algorithm for constructing the set of generators in the nondegenerate case, and left the degenerate case for future work. In their recent work, Kolda *et al.* [14] mentioned that the problem when constraints are degenerate has been well studied in computational geometry and the solution to the problem exists in [5, 6]. However, there are examples when the method proposed in [5, 6] requires full enumeration, which can be cost-prohibitive.

Price and Coope [21] gave as an aside a result that can be used for constructing a set of generators in the degenerate case. It follows from their result that, in order to construct a set of generators, it is sufficient to consider maximal linearly independent subsets of the active constraints. However, this approach implies enumeration of all possible linearly independent subsets of maximal rank and does not take into account properties of the problem that can help to reduce this enumeration. Price and Coope [21] outlined an algorithm for constructing frames, but it was not their point to work out details of the numerical implementation in the degenerate case.

The purpose of this paper is to give detailed consideration to GPS in the degenerate case in a way that is complementary to [4] and [16]. Our main result is a detailed algorithm for constructing the set of generators at a current GPS iterate in both the degenerate and nondegenerate cases. To construct the set of generators in the degenerate case, we identify the redundant and nonredundant active constraints and then use either QR decomposition or a construction proposed in [16].

Classification of constraints as redundant or nonredundant is one of the main issues here, because it is sufficient to construct the set of generators only for nonredundant constraints. Several methods for classifying constraints exist. For example, there are deterministic algorithms [10, 13], probabilistic hit-and-run methods [7], and a probabilistic method based on an equivalence between the constraint classification problem and the problem of finding a feasible solution to a set covering problem [9]. A survey and comparison of strategies for classifying constraints are given in [9, 13]. Any of these approaches can be applied in the GPS framework to identify redundant and nonredundant constraints. However, in the paper, we propose a new projection approach to identify nonredundant constraints that is more suitable for GPS methods.

The projection method is similar to the hit-and-run algorithm [7], in which nonredundant constraints are searched for along random direction vectors from each point in a sequence of random interior points, but differs in its use of a deterministic direction. The major advantage of the projection method for our application is that the number of direction vectors (in the terminology of the hit-and-run algorithm) is equal to the number of constraints that have to be identified. For us this is generally a small number. In the hit-and-run algorithm, this number is determined by a stop criterion and can be large if many of the randomly generated directions do not detect a nonredundant constraint. Moreover, the formulas used in the projection method are simpler than those used for computing the intersection points of a direction vector with the hyperplanes in the hit-and-run algorithm. We should note also that the goal of hit-and-run is to detect all nonredundant constraints in a full system of linear inequalities. We use the projection method to detect the nonredundant constraints among only active constraints in the case when they are linearly dependent. As our numerical tests show, the projection method cheaply detects all, or almost all, nonredundant constraints.

To classify constraints not detected by the projection method, we use another approach outlined in [10].

As a result, we ensure that every active constraint is detected as either redundant or nonredundant. In the worst case, we may have linearly dependent, nonredundant constraints. We propose a general approach for handling this case with an accompanying convergence theorem, along with two specific instances that can be used effectively in practice.

In the end, we briefly discuss the extension of our ideas to optimization problems with general nonlinear constraints that are linearly dependent at a solution. We do so by applying the projection method to a linearization of the constraints, and we argue that it is less costly than applying the approach of [10].

The organization of the paper is as follows. In the next section, we give a brief description of GPS as well as the convergence result for linearly constrained minimization following papers by Audet and Dennis [4], and by Lewis and Torczon [16]. Section 3 is devoted to the topic of redundancy. In the first part of the section, we introduce a definition of the ε-active constraints and discuss some scaling issues. The second part of Section 3 contains essential definitions and results on redundancy [10, 13, 19, 25] that are required for our analysis. Then we propose our projection method to determine nonredundant constraints, and we briefly describe a more expensive follow-up approach to be applied if some constraints are not identified by the projection method. In Section 4, we give an algorithm for constructing the set of generators and discuss implementation details, including a new approach for handling nonredundant linearly dependent constraints in a rigorous way without significantly increasing computational cost. In Section 5, we consider the extension of our ideas to nonlinearly constrained problems. Section 6 is devoted to some concluding remarks.

Notation. $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{N}$ denote the set of real numbers, integers, and nonnegative integers, respectively. For any finite set S, we may refer to the matrix S as the one whose columns are the elements of S. Similarly, for any matrix A, the notation $a \in A$ means that a is a column of A.

2. Generalized pattern search algorithms. In this section, we briefly describe the class of GPS algorithms for linearly constrained minimization, along with the main convergence result. We follow papers by Audet and Dennis [4] and by Lewis and Torczon [16], and we refer the reader there for details of managing the mesh size Δ_k. Throughout, we will always use the ℓ_2 norm.

GPS algorithms can be applied either to the objective function f or to the barrier function $f_\Omega = f + \psi_\Omega : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, where ψ_Ω is the indicator function for Ω, which is zero on Ω and ∞ elsewhere. The value of f_Ω is $+\infty$ on all points that are either infeasible or at which f is declared to be $+\infty$. This barrier approach is probably as old as direct search methods themselves.

A GPS algorithm for linearly constrained optimization generates a sequence of iterates $\{x_k\}$ in Ω. The *current iterate* $x_k \in \mathbb{R}^n$ is chosen from a finite number of points on a *mesh*, which is a discrete subset of $\mathbb{R}^n$. At iteration k, the mesh is centered around the *current mesh point (current iterate)* x_k and its fineness is parameterized through the *mesh size parameter* $\Delta_k > 0$ as

$$M_k = \{x_k + \Delta_k D z \,:\, z \in \mathbb{N}^{n_D}\}, \tag{2.1}$$

where D is a finite matrix whose columns form a *set of positive spanning directions* in $\mathbb{R}^n$, n_D is the number of columns of the matrix D. At each iteration, some positive spanning matrix D_k composed of columns of D is used to construct the *poll set*,

$$P_k = \{x_k + \Delta_k d \,:\, d \in D_k\}. \tag{2.2}$$

A two-dimensional mesh and poll set are illustrated in Figure 2.1.

If $x_k \in \Omega$ is not near the boundary, then D_k is a positive spanning set for $\mathbb{R}^n$ [16]. If $x_k \in \Omega$ is near the boundary, the matrix D_k is constructed so its columns d_j also span the cone of feasible directions at x_k and conform to the geometry of the boundary of Ω. Hence, the set D must be rich enough to contain generators

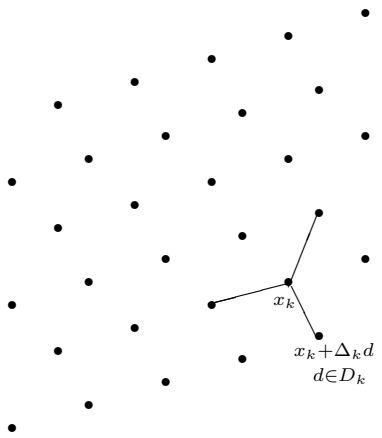

FIG. 2.1. *A mesh and poll set in* $\mathbb{R}^2$

for the tangent cone $T_\Omega(x) = \text{cl}\{\mu(\omega - x) : \mu \geq 0,\ \omega \in \Omega\}$ for every $x \in \Omega$. More formally, the sets D_k must satisfy the following definition.

DEFINITION 2.1. *A rule for selecting the positive spanning sets* $D_k \subseteq D$ conforms *to* Ω *for some* $\varepsilon > 0$, *if at each iteration* k *and for each* y *in the boundary of* Ω *for which* $\|y - x_k\| < \varepsilon$, $T_\Omega(y)$ *is generated by a nonnegative linear combination of columns of* D_k.

Each GPS iteration is divided into two phases: an optional search and a local poll. In each step, the barrier objective function is evaluated at a finite number of mesh points in an attempt to find one that yields a lower objective function value than the incumbent. We refer to such a point as an *improved mesh point*. If an improved mesh point is found, it becomes the incumbent, so that $f(x_{k+1}) < f(x_k)$. The mesh size parameter is then either held constant or increased.

In the SEARCH step, there is complete flexibility. Any strategy may be used (including none), and the user's knowledge of the domain may be incorporated. If the SEARCH step fails to yield an improved mesh point, the POLL step is invoked. In this second step, the barrier objective function is evaluated at points in the poll set P_k (*i.e.*, neighboring mesh points) until an improved mesh point is found or until all the points in P_k have been evaluated. If both the SEARCH and POLL steps fail to find an improved mesh point, then the incumbent is declared to be a *mesh local optimizer* and is retained as the incumbent, so that $x_{k+1} = x_k$. The mesh size parameter is then decreased. Figure 2.2 gives a description of a basic GPS algorithm.

We remind the reader that the normal cone $N_\Omega(x)$ to Ω at x is the nonnegative span of all the outwardly pointing constraint normals at x and can be written as the polar of the tangent cone: $N_\Omega(x) = \{v \in \mathbb{R}^n : \forall \omega \in T_\Omega(x),\ v^T\omega \leq 0\}$.

Assumptions. We make the following standard assumptions [4]:

A1 A function f_Ω and $x_0 \in \mathbb{R}^n$ (with $f_\Omega(x_0) < \infty$) are available.
A2 The constraint matrix A is rational.
A3 All iterates $\{x_k\}$ produced by the GPS algorithm lie in a compact set.

Under these assumptions, Torczon [22] showed that $\liminf \Delta_k = 0$, and Audet and Dennis [4] identified the following subsequences, for which the limit of Δ_k is zero.

DEFINITION 2.2. *A subsequence of mesh local optimizers* $\{x_k\}_{k\in K}$ *(for some subset of indices* K*) is said to be a* refining subsequence *if* $\{\Delta_k\}_{k\in K}$ *converges to zero.*

Audet and Dennis [4] proved the following convergence results for GPS in the linearly constrained case

- INITIALIZATION:
  Let x_0 be such that $f_\Omega(x_0)$ is finite. Let D be a positive spanning set, and let M_0 be the mesh on $\mathbb{R}^n$ defined by $\Delta_0 > 0$, and D_0. Set the iteration counter $k = 0$.
- SEARCH AND POLL STEP:
  Perform the SEARCH and possibly the POLL steps (or only part of them) until an improved mesh point x_{k+1} with the lowest f_Ω value so far is found on the mesh M_k defined by equation (2.1).
  - Optional SEARCH: Evaluate f_Ω on a finite subset of trial points on the mesh M_k defined by (2.1) (the strategy that gives the set of points is usually provided by the user; it must be finite and the set can be empty).
  - Local POLL: Evaluate f_Ω on the poll set defined in (2.2).
- PARAMETER UPDATE:
  If the SEARCH or the POLL step produced an improved mesh point, *i.e.*, a feasible iterate $x_{k+1} \in M_k \cap \Omega$ for which $f_\Omega(x_{k+1}) < f_\Omega(x_k)$, then update $\Delta_{k+1} \geq \Delta_k$.
  Otherwise, $f_\Omega(x_k) \leq f_\Omega(x_k + \Delta_k d)$ for all $d \in D_k$ and so x_k is a mesh local optimizer. Set $x_{k+1} = x_k$, update $\Delta_{k+1} < \Delta_k$.
  Increase $k \leftarrow k + 1$ and go back to the SEARCH and POLL step.

FIG. 2.2. *A simple GPS algorithm*

using only these assumptions.

LEMMA 2.3. *Under assumptions A1–A3, if $\hat{x}$ is any limit of a refining subsequence, if d is any direction in D for which f at a* POLL *step was evaluated for infinitely many iterates in the subsequence, and if f is Lipschitz near $\hat{x}$, then the generalized directional derivative of f at $\hat{x}$ in the direction d is nonnegative, i.e., $f^\circ(\hat{x}; d) \geq 0$.*

THEOREM 2.4 (Convergence to a Karush-Kuhn-Tucker point). [4] *Under assumptions A1–A3, if f is strictly differentiable at a limit point $\hat{x}$ of a refining subsequence, and if the rule for selecting positive spanning sets $D_k \subseteq D$ conforms to Ω for some $\varepsilon > 0$, then $\nabla f(\hat{x})^T \omega \geq 0$ for all $\omega \in T_\Omega(\hat{x})$, and so $-\nabla f(\hat{x}) \in N_\Omega(\hat{x})$. Thus, $\hat{x}$ is a Karush-Kuhn-Tucker point.*

The purpose of this paper is to provide an algorithm for constructing sets D_k that conform to the boundary of Ω. If the active constraints are linearly dependent, we apply strategies for the identification of redundant and nonredundant constraints, which are described in the next section, and then construct sets D_k taking into account only nonredundant constraints. We now pause to outline the main results concerning redundancy from mathematical programming, and then in Section 4, we continue consideration of GPS and strategies for constructing the sets D_k.

3. Redundancy. We now present some essential definitions and results concerning redundancy [7, 10, 13, 19, 25] that are required for our analysis. Then we propose our approach, the projection method, to determining the nonredundant constraints and briefly describe another approach that is applied if some constraints are not identified by the projection method.

We consider the feasible region Ω defined by (1.2), and refer to the inequality $a_j^T x \leq b_j$ as the *j-th constraint*. The region represented by all but the jth constraint is given by

$$\Omega_j = \{x \in \mathbb{R}^n \, : \, a_i^T x \leq b_i, \; i \in I \backslash \{j\}\},$$

where $I \backslash \{j\}$ is the set I with the element j removed.

The following definition is consistent with definitions given in [10, 13].

DEFINITION 3.1 (Redundant constraint). *The jth constraint $a_j^T x \le b_j$ is* redundant *in the description of Ω if and only if $\Omega = \Omega_j$, and is (necessarily)* nonredundant *otherwise.*

3.1. ε–active constraints. We next compare two definitions of ε–active constraints and discuss some associated scaling issues. They are replicated from [16] and [21], respectively.

DEFINITION 3.2. (*e.g.,* [21]). *Let some scalar $\varepsilon > 0$ be given and $x_k \in \Omega$. The jth constraint is ε–active at x_k if*

$$0 \le b_j - a_j^T x_k \le \varepsilon. \tag{3.1}$$

DEFINITION 3.3. (*e.g.,* [16]). *Let some scalar $\varepsilon > 0$ be given and $x_k \in \Omega$. The jth constraint is ε–active at x_k if*

$$\operatorname{dist}(x_k, H_j) \le \varepsilon, \tag{3.2}$$

where $H_j = \{x \in \mathbb{R}^n : a_j^T x = b_j\}$, and $\operatorname{dist}(x_k, H_j) = \min_{y \in H_j} \|y - x_k\|$ is the distance from x_k to the hyperplane H_j.

Clearly, the jth constraint can be made ε–active at x_k in the sense of Definition 3.2 by multiplying the inequality $b_j - a_j^T x_k \ge 0$ by a sufficiently small number. On the other hand, this multiplication does not change the distance between the point x_k and any H_j defined in Definition 3.3. In the paper, we prefer to use Definition 3.2, since it is easier to check than Definition 3.3. However, Definition 3.2 is proper, if we assume preliminary scaling of the constraints so that the following lemma applies.

LEMMA 3.4. *Let some scalar $\varepsilon > 0$ be given, $x_k \in \Omega$, and $\|a_j\| = 1$ for all $j \in I$ in* (1.2)*. Then, for any $j \in I$, Definition* 3.2 *of the ε–active constraint is equivalent to Definition* 3.3*, and the projection $P_j(x_k)$ of the point x_k onto the hyperplane $H_j = \{x \in \mathbb{R}^n : a_j^T x = b_j\}$ is defined by*

$$P_j(x_k) = x_k + a_j(b_j - a_j^T x_k). \tag{3.3}$$

Proof. For any $j \in I$, the distance from x_k to the hyperplane H_j is given by

$$\operatorname{dist}(x_k, H_j) = \frac{|b_j - a_j^T x_k|}{\|a_j\|}. \tag{3.4}$$

Hence, if $\|a_j\| = 1$ and $x_k \in \Omega$, (3.1) is equivalent to (3.2).

By definition of the projection of x_k onto H_j,

$$\|P_j(x_k) - x_k\| = \operatorname{dist}(x_k, H_j).$$

Since $x_k \in \Omega$ and $\|a_j\| = 1$, it follows from (3.4) that $\operatorname{dist}(x_k, H_j) = b_j - a_j^T x_k$ and

$$P_j(x_k) = x_k + a_j \operatorname{dist}(x_k, H_j) = x_k + a_j(b_j - a_j^T x_k).$$

Hence, (3.3) holds. ∎

To satisfy the conditions of Lemma 3.4, we introduce the matrix $\bar{A}$ and vector $\bar{b}$ that are additional scaled copies of A and b, respectively from (1.2), such that

$$\bar{a}_i = \frac{a_i}{\|a_i\|}, \quad \bar{b}_i = \frac{b_i}{\|a_i\|}, \quad i \in I. \tag{3.5}$$

Consequently, $\|\bar{a}_i\| = 1$ for all $i \in I$ and $\Omega = \{x \in \mathbb{R}^n : A^T x \leq b\} = \{x \in \mathbb{R}^n : \bar{A}^T x \leq \bar{b}\} = \{x \in \mathbb{R}^n : \bar{a}_i^T x \leq \bar{b}_i,\ i \in I\}$.

We then use $\bar{A}$ and $\bar{b}$ to define the set of indices of the ε–active constraints as

$$I(x_k, \varepsilon) = \{i \in I : 0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon\}, \tag{3.6}$$

and we apply the projection method for detection of the nonredundant constraints (see Section 3.3.1 for more details.) We refer to the set $I(x_k, \varepsilon)$ as the *working index set* at the current iterate x_k.

This paper also makes use of the regions given by

$$\Omega(x_k, \varepsilon) = \{x \in \mathbb{R}^n : a_i^T x \leq b_i,\ i \in I(x_k, \varepsilon)\}, \tag{3.7}$$

and

$$\Omega_j(x_k, \varepsilon) = \{x \in \mathbb{R}^n : a_i^T x \leq b_i,\ i \in I(x_k, \varepsilon) \backslash \{j\}\}, \quad j \in I(x_k, \varepsilon).$$

Clearly, $\Omega \subseteq \Omega(x_k, \varepsilon) \subseteq \Omega_j(x_k, \varepsilon)$. Furthermore, since $\Omega \subseteq \Omega(x_k, \varepsilon)$, if the jth constraint is redundant in the description of $\Omega(x_k, \varepsilon)$, it is also redundant in the description of Ω.

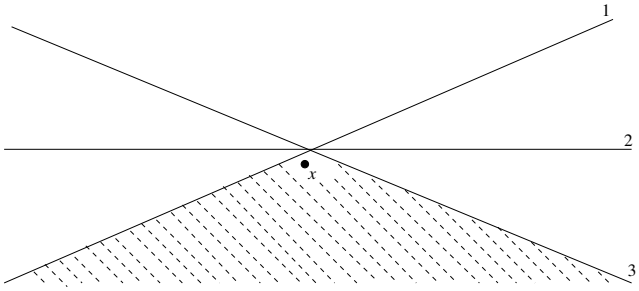

FIG. 3.1. *An illustration of ε-active and redundant constraints. Constraints* 1, 2, *and* 3 *are ε-active at the current iterate x and constraint* 2 *is redundant.*

3.2. Redundancy in mathematical programming. We now give definitions and theorems consistent with the mathematical programming literature [7, 10, 13, 19, 25]. We begin with the following definitions, which can be found in [19, 25]. In the discussion that follows, we use notation consistent with that of Section 1 (see (1.2) and the discussion that follows it).

DEFINITION 3.5 (Polyhedron). *A subset of $\mathbb{R}^n$ described by a finite set of linear constraints $P = \{x \in \mathbb{R}^n : C^T x \leq d\}$ is a polyhedron.*

Obviously, Ω given by (1.2) and $\Omega(x_k, \varepsilon)$ given by (3.7) are polyhedra.

DEFINITION 3.6. *The points $x^1, \ldots, x^k \in \mathbb{R}^n$ are* affinely independent *if the $k-1$ directions $x^2 - x^1, \ldots, x^k - x^1$ are linearly independent, or alternatively, the k vectors $(x^1, 1), \ldots, (x^k, 1) \in \mathbb{R}^{n+1}$ are linearly independent.*

We will assume that Ω is full-dimensional, as defined below.

DEFINITION 3.7. *The dimension of P, denoted* $\dim(P)$*, is one less than the maximum number of affinely independent points in P. Then $P \subseteq \mathbb{R}^n$ is* full-dimensional *if and only if* $\dim(P) = n$.

Note that, if Ω were not full-dimensional, then a barrier GPS approach would not be a reasonable way to handle linear constraints because it would be difficult to find any trial in Ω. Since we assume Ω is

full-dimensional, this implies that its supersets $\Omega(x_k, \varepsilon)$ and $\Omega_j(x_k, \varepsilon)$ are full-dimensional.

DEFINITION 3.8 (Valid inequality). *An inequality $c_j^T x \le d_j$ is a valid inequality for $P \subseteq \mathbb{R}^n$ if $c_j^T x \le d_j$ for all $x \in P$.*

DEFINITION 3.9 (Face and Facet). (i) *F defines a face of the polyhedron P if $F = \{x \in P : c_j^T x = d_j\}$ for some valid inequality $c_j^T x \le d_j$ of P. $F \neq \emptyset$ is said to be a proper face of P if $F \neq P$.*
(ii) *F is a facet of P if F is a face of P and* $\dim(F) = \dim(P) - 1$.

DEFINITION 3.10 (Interior point). *A point $x \in P$ is called an interior point of P if $C^T x < d$.*

We also need the following results from integer programming [25, pp. 142–144] and [19, pp. 85–92].

PROPOSITION 3.11. [19, Corollary 2.5] *A polyhedron is full-dimensional if and only if it has an interior point.*

THEOREM 3.12. [25, Theorem 9.1] *If P is a full-dimensional polyhedron, it has a unique minimal description*

$$P = \{x \in \mathbb{R}^n : c_i^T x \le d_i, \quad i = 1, \ldots, m\},$$

where each inequality is unique to within a positive multiple.

COROLLARY 3.13. [25, Proposition 9.2] *If P is full-dimensional, a valid inequality $c_j^T x \le d_j$ is necessary in the description of P if and only if it defines a facet of P.*

Corollary 3.13 means that the following concepts are equivalent for $\Omega(x_k, \varepsilon)$ defined in (3.7).

- The jth inequality $a_j^T x \le b_j$ defines a facet of $\Omega(x_k, \varepsilon)$.
- The jth inequality $a_j^T x \le b_j$ is necessary (nonredundant) in description of $\Omega(x_k, \varepsilon)$, or in other words,

$$\Omega(x_k, \varepsilon) \subsetneq \Omega_j(x_k, \varepsilon). \tag{3.8}$$

Our approach for identifying nonredundant constraints is based primarily on the following proposition.

PROPOSITION 3.14. *Let a working index set $I(x_k, \varepsilon)$ be given. An inequality $a_j^T x \le b_j$, $j \in I(x_k, \varepsilon)$, is nonredundant in the description of $\Omega(x_k, \varepsilon)$ if and only if either $I(x_k, \varepsilon) = \{j\}$ or there exists $\bar{x} \in \mathbb{R}^n$ such that $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \backslash \{j\}$.*

Proof. Since the case $I(x_k, \varepsilon) = \{j\}$ is trivial, we give the proof for the case when $I(x_k, \varepsilon) \backslash \{j\} \neq \emptyset$.

Necessity. Since the inequality $a_j^T x \le b_j$ is nonredundant, then, by (3.8), there exists $x^* \in \mathbb{R}^n$ such that $a_i^T x^* \le b_i$ for all $i \in I(x_k, \varepsilon) \backslash \{j\}$, and $a_j^T x^* > b_j$. By Proposition 3.11, there exists an interior point $\hat{x} \in \Omega(x_k, \varepsilon)$ such that $a_i^T \hat{x} < b_i$ for all $i \in I(x_k, \varepsilon)$. Thus on the line between x^* and $\hat{x}$ there is a point $\bar{x} \in \mathbb{R}^n$ satisfying $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \backslash \{j\}$.

Sufficiency. Let $\hat{x} \in \Omega(x_k, \varepsilon)$ be an interior point, *i.e.*, $a_i^T \hat{x} < b_i$ for all $i \in I(x_k, \varepsilon)$. Since there exists $\bar{x} \in \mathbb{R}^n$ such that $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \backslash \{j\}$, then there exists $\delta > 0$ such that $\tilde{x} = \bar{x} + \delta(\bar{x} - \hat{x})$ satisfies $a_j^T \tilde{x} > b_j$ and $a_i^T \tilde{x} \le b_i$, $i \in I(x_k, \varepsilon) \backslash \{j\}$. Therefore, (3.8) holds, and by Definition 3.1, the jth constraint is nonredundant. □

Proposition 3.14 means that if the jth constraint, $j \in I(x_k, \varepsilon)$, is nonredundant, then there exists a feasible point $\bar{x} \in \Omega(x_k, \varepsilon)$ such that only this constraint holds with equality at $\bar{x}$.

Our approach for identifying redundant constraints is based primarily on the following theorem [10].

THEOREM 3.15. *The jth constraint is redundant in system* (1.2) *if and only if the linear program,*

$$\text{maximize } a_j^T x, \quad \text{subject to } x \in \Omega_j, \tag{3.9}$$

has an optimal solution x^* such that $a_j^T x^* \le b_j$.

3.3. Approaches for identifying redundant and nonredundant constraints. We now outline two approaches for identifying redundancy in the constraint set: a projection method for identifying nonredundant constraints and a linear programming (LP) approach for identifying redundant ones. The LP approach, which is based on Theorem 3.15, is described in [10]. In Section 4, we will explain in more detail how these ideas are implemented in the class of GPS algorithm for linearly constrained problems, even in the presence of degeneracy.

3.3.1. A projection method. The main idea of the projection method we propose is the construction, if possible, of a point $\bar{x}$ such that $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \backslash \{j\}$. If such a point $\bar{x}$ exists, then by Proposition 3.14, the jth constraint is nonredundant.

Recall that we defined in (3.5) a scaled copy $\bar{A}$ of the matrix A and a scaled vector $\bar{b}$. We denote by $P_j(x_k)$, the projection of $x_k \in \mathbb{R}^n$ onto the hyperplane $H_j = \{x \in \mathbb{R}^n : \bar{a}_j^T x = \bar{b}_j\}$. Assume that $x_k \in \Omega$. Then by (3.3) and by $\|\bar{a}_j\| = 1$,

$$P_j(x_k) = x_k + \bar{a}_j(\bar{b}_j - \bar{a}_j^T x_k). \tag{3.10}$$

The following proposition is the main one for the projection method.

PROPOSITION 3.16. *Let $x_k \in \Omega$ and let a working index set $I(x_k, \varepsilon)$ be given. An inequality $a_j^T x \le b_j$, $j \in I(x_k, \varepsilon)$, is nonredundant in the description of $\Omega(x_k, \varepsilon)$ if*

$$\bar{a}_i^T P_j(x_k) < \bar{b}_i \quad \text{for all} \quad i \in I(x_k, \varepsilon) \backslash \{j\}, \tag{3.11}$$

where $P_j(x_k)$ is a projection of x_k onto H_j.

Proof. The proof follows from Proposition 3.14. □

Proposition 3.16 allows us to very quickly classify the jth constraint as nonredundant if (3.11) holds for all $i \in I(x_k, \varepsilon) \backslash \{j\}$, where $P_j(x_k)$ in (3.11) is obtained from (3.10). The only drawback is that it identifies nonredundant constraints and not redundant ones.

3.3.2. The linear programming approach. If some constraints have not been identified by the projection method, we can apply another approach based on Theorem 3.15 to identify redundant and nonredundant constraints. It follows from Theorem 3.15 that all redundant and nonredundant constraints could be conclusively identified by solving n LP problems of the form given in (3.9). While doing so is clearly more expensive than the projection method given in Section 3.3.1, it could be accomplished during the initialization step of GPS (*i.e.*, before the GPS iteration sequence begins), at a cost of solving n LP problems. This is possible because redundancy of linear constraints is independent of the location of the current iterate. However, the projection method could be advantageous when many linear constraints are present (which is often the case with redundant constraints), or when dealing with linear constraints formed by linearizing nonlinear ones. In the latter case, redundancy would depend upon location in the domain, since the linear constraints would change based on location.

The book [13] describes different methods in the context of the LP approach. They include some very special propositions involving slack variables that simplify and reduce the computational cost of the numerical solution of the LP problem (3.9). We refer the reader to [13] for a more detailed discussion of these issues.

4. Construction of the set of generators. The purpose of this section is to provide a detailed algorithm for constructing the set of directions D_k introduced in Section 2, even in the presence of degenerate constraints.

Let some scalar $\varepsilon > 0$ be given, and let $\bar{a}_i^T$ be the ith row of the matrix $\bar{A}^T$ in (3.5). At the current iterate x_k, we construct the working index set $I(x_k, \varepsilon)$ such that

$$0 \le \bar{b}_i - \bar{a}_i^T x_k \le \varepsilon \iff i \in I(x_k, \varepsilon).$$

The last inequality means that every constraint that is active at x_k or at some point near x_k appears in $I(x_k, \varepsilon)$. In [4], the authors suggest not setting ε so small that Δ_k is made small by approaching the boundary too closely before including conforming directions that allow the iterates to move along the boundary of Ω. A good discussion of how to choose ε can be found in [14].

Without loss of generality, we assume that $I(x_k, \varepsilon) = \{1, \ldots, m\}$, for $m \geq 2$. This avoids more cumbersome notation, like $I(x_k, \varepsilon) = \{i_1(x_k, \varepsilon), \ldots, i_m(x_k, \varepsilon)\}$. Furthermore, we denote by B_k, the matrix whose columns are the columns of A corresponding to the indices $I(x_k, \varepsilon) = \{1, \ldots, m\}$; *i.e.*,

$$B_k = [a_1, \ldots, a_m]. \tag{4.1}$$

4.1. Classification of degeneracy at the current iterate. Let the matrix B_k be defined by (4.1). At the current iterate x_k, the matrix B_k satisfies one of the following conditions:

- *nondegenerate case*: B_k has full rank;
- *degenerate redundant case*: B_k does not have full rank and the nonredundant constraints are linearly independent;
- *degenerate nonredundant case*: B_k does not have full rank and the nonredundant constraints are linearly dependent.

The last condition is illustrated by following example provided to us by Charles Audet.

Example 2. Suppose that the feasible region Ω (see (1.2)), shown in Figure 4.1, is defined by the following system of inequalities:

$$\begin{array}{lcl} x_1 - 2x_2 - 2x_3 & \leq & 0 \\ -2x_1 + x_2 - 2x_3 & \leq & 0 \\ -2x_1 - 2x_2 + x_3 & \leq & 0 \\ x_1 & \geq & 0 \\ x_2 & \geq & 0 \\ x_3 & \geq & 0 \end{array} \tag{4.2}$$

If $x_k \in \mathbb{R}^3$ is near the origin, all six constraints are active, linearly dependent, and nonredundant. The matrix B_k is given as

$$B_k = \begin{pmatrix} 1 & -2 & -2 & -1 & 0 & 0 \\ -2 & 1 & -2 & 0 & -1 & 0 \\ -2 & -2 & 1 & 0 & 0 & -1 \end{pmatrix}.$$

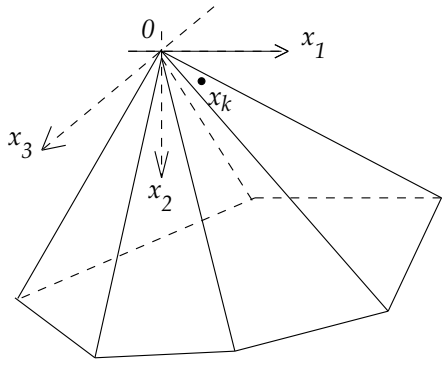

FIG. 4.1. *Example 2. An illustration of the degenerate nonredundant case.*

4.2. Set of generators. Following [16], we define the cone $K(x_k, \varepsilon)$ as the cone generated by the normals to the ε–active constraints, and $K^\circ(x_k, \varepsilon)$ as its polar:

$$K^\circ(x_k, \varepsilon) = \{w \in \mathbb{R}^n : a_i^T w \leq 0 \quad \forall i \in I(x_k, \varepsilon)\}. \tag{4.3}$$

This cone can also be expressed as a finitely generated cone [24]. To see this, first consider the following definition.

DEFINITION 4.1 (Set of generators). *A set $V = \{v_1, \ldots, v_r\}$ is called a set of generators of the cone K defined by* (4.3) *if the following conditions hold:*

1. *Every vector $v \in K$ can be expressed as a nonnegative linear combination of vectors in V.*
2. *No proper subset of V satisfies 1.*

Thus, given Definition 4.1, we can express $K^\circ(x_k, \varepsilon)$ as

$$K^\circ(x_k, \varepsilon) = \{w \in \mathbb{R}^n : w = \sum_{j=1}^{r} \lambda_j v_j, \quad \lambda_j \geq 0, v_j \in \mathbb{R}^n, \quad j = 1, \ldots, r\}, \tag{4.4}$$

where the $V = \{v_1, \ldots, v_r\}$ is the set of generators for $K^\circ(x_k, \varepsilon)$.

The key idea, which was first suggested by May in [18] and applied to GPS in [16], is to include in D_k the generators of the cone $K^\circ(x_k, \varepsilon)$. Hence, the problem of construction of the set D_k reduces to the problem of constructing generators $\{v_1, \ldots, v_r\}$ of the cone $K^\circ(x_k, \varepsilon)$ and then completing them to a positive spanning set for $\mathbb{R}^n$.

The following proposition means that it is sufficient to construct the set of generators only for nonredundant constraints.

PROPOSITION 4.2. *Let $I(x_k, \varepsilon)$ be the set of indices of ε–active constraints at $x_k \in \mathbb{R}^n$. Let $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$ be the subset of indices of the nonredundant constraints that define $\Omega(x_k, \varepsilon)$. Let the cone $K^\circ(x_k, \varepsilon)$ be defined by* (4.3) *and let the cone $K_N^\circ(x_k, \varepsilon)$ be given by*

$$K_N^\circ(x_k, \varepsilon) = \{w \in \mathbb{R}^n : a_i^T w \leq 0 \quad \forall i \in I_N(x_k, \varepsilon)\}.$$

If $\{v_1, \ldots, v_p\}$ is a set of generators for $K_N^\circ(x_k, \varepsilon)$, then it is also a set of generators for $K^\circ(x_k, \varepsilon)$.

Proof. The proof of this proposition follows from Corollary 3.13. □

Pattern search methods require that iterates lie on a rational lattice [16]. To ensure this, Lewis and Torczon [16] placed an additional requirement that the matrix of constraints A^T in (1.2) is rational. Under this requirement, Lewis and Torczon [16] showed, in the following theorem, that it is always possible to find rational generators for the cones $K^\circ(x_k, \varepsilon)$, which, with the rational mesh size parameter Δ_k, ensures that GPS iterates lie on a rational lattice.

THEOREM 4.3. *If K is a cone with rational generators V, then there exists a set of rational generators for K°.*

Moreover, for the case of linearly independent active constraints, Lewis and Torczon [16] proposed constructing the set of generators for all the cones $K^\circ(x_k, \varepsilon)$, $0 \leq \varepsilon \leq \delta$, as follows:

THEOREM 4.4. *Suppose that for some δ, $K(x, \delta)$ has a linearly independent set of rational generators V. Let N be a rational positive basis for the null space of V^T. Then, for any ε, $0 \leq \varepsilon \leq \delta$, a set of rational generators for $K^\circ(x, \varepsilon)$ can be found among the columns of N, $V(V^TV)^{-1}$, and $-V(V^TV)^{-1}$.*

The matrix N can be constructed by taking columns of the matrices $\pm(I - V(V^TV)^{-1}V^T)$ [16].

Recall that we use the scaled matrix $\bar{A}$ defined in (3.5) to determine ε-active, redundant, and nonredundant constraints. Then we use the result stated in Theorem 4.4 together with rational columns of A, which correspond to the nonredundant and ε-active constraints, to obtain a set of rational generators.

A set of generators, which may be irrational in exact arithmetic, can also be found by using the QR factorization of the matrix V. The following corollary shows how to use the QR factorization of V to construct the generators for all the cones $K^\circ(x_k, \varepsilon)$, $0 \leq \varepsilon \leq \delta$. Recall that the full QR factorization of V can be represented as

$$V = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}, \tag{4.5}$$

where R_1 is upper triangular and $\text{rank}(R_1) = \text{rank}(V)$, and the columns of Q_1 form an orthonormal basis for the space spanned by the columns of V, while the columns of Q_2 constitute an orthonormal basis for null space of V^T.

COROLLARY 4.5. *Suppose that for some δ, $K(x, \delta)$ has a linearly independent set of rational generators V. Then, for any ε, $0 \leq \varepsilon \leq \delta$, a set of generators for $K^\circ(x, \varepsilon)$ can be found among the columns of Q_2, $Q_1 R_1 (R_1^T R_1)^{-1}$, and $-Q_1 R_1 (R_1^T R_1)^{-1}$.*

Proof. By substituting $V = QR$ and using the properties of the matrices in the QR factorization, we obtain

$$V(V^T V)^{-1} = QR((QR)^T (QR))^{-1} = QR(R^T Q^T QR)^{-1} = QR(R^T R)^{-1}. \tag{4.6}$$

By applying Theorem 4.4 and by taking into account that columns of Q_2 span the null space of V^T, we obtain the statement of the corollary. □

From the theoretical point of view, a set of generators obtained by using Corollary 4.5 may be irrational since an implementation of the QR decomposition involves calculation of square roots. This would violate theoretical assumptions required for convergence of pattern search. However, since V is rational, both sides of (4.6) must also be rational. Therefore, by Corollary 4.5, any generators with irrational elements would be found in the matrix Q_2. But in the degenerate case, Q_2 will often be empty, since it represents a positive spanning set for the null space of V^T, and most examples of degeneracy occur when the number of ε-active constraints exceeds the number of variables. Furthermore, since we use floating point arithmetic in practice, irrational generators would be represented as rational approximations. This has the effect of generating a slightly different cone. Thus, it would be enough to ensure convergence, but to a stationary point of a slightly different problem. However, the error experienced in representing an irrational number as rational is probably smaller than the typical roundoff error associated with LU factorization.

4.3. The nonredundant degenerate case. Perhaps the most difficult case to handle is the one in which the ε-active constraints at x_k are nonredundant, but linearly dependent. This can happen, in particular, when there are more ε-active constraints than variables, as is the case in Example 4.2. The difficulty of this case lies in the fact that the number of directions required to generate the tangent cone can become large.

Let $S_k = \{a_1, a_2, \ldots, a_p\}$ denote the set of vectors corresponding to the ϵ-active nonredundant constraints at x_k. Price and Coope [21] showed that, in order to construct D_k, it is sufficient to identify the tangent cone generators of all maximally linearly independent subsets of S_k. For S_k with $r = \text{rank}(S_k)$, we can estimate the number s of these subsets, by

$$s = \frac{p!}{r!(p-r)!}. \tag{4.7}$$

Thus, in order to identify the entire set of tangent cone generators, we would have to consider s different sets of positive spanning directions, where s could become quite large. While there are some vertex enumeration techniques [5] (mentioned in [16]) that could be useful, we now present a more practical approach – first in general, and then followed by some specific instances that can be implemented in practice.

4.3.1. Partially conforming generator sets. In our approach, we choose a subset of r linearly independent elements of S_k and store them as columns of B_k. Based on the methods described in Section 3.3.1, we can construct a set of generators for the cone defined only by a subset of the constraints represented in S_k. Furthermore, we require B_k to change at each iteration so that, in the limit, each constraint that is active at the limit point $\hat{x}$ has been used infinitely often in constructing directions. Since the set of tangent cone generators is finite, the ordering scheme for ensuring this is straightforward.

This approach is essentially equivalent to using all the tangent cone generators, except that it is spread out over more than one iteration. The advantage is that it keeps the size of the poll set no larger than it would be in the nondegenerate case. However, the drawback is that we no longer have a full set of directions that conform to the geometry of Ω, which is an important hypothesis in the statement of Theorem 2.4.

The proof of Theorem 2.4, given in [4], relies on two crucial ideas; namely, Lemma 2.3 and the use of conforming directions. Under the proposed method of handling degenerate nonredundant constraints, Lemma 2.3 still applies, but Theorem 2.4 cannot be applied, since not all the tangent cone generators are used at each iteration. We introduce the following theorem, which establishes the same result as Theorem 2.4, but with a different hypothesis and a proof that is essentially identical (see [4]).

THEOREM 4.6. *Let $\hat{x} \in \Omega$ be the limit point of a refining subsequence $\{x_k\}_{k\in K}$. Under Assumptions A1–A3, if f is strictly differentiable at $\hat{x}$ and all generators of the tangent cone $T_\Omega(\hat{x})$ are used infinitely often in K, then $\nabla f(\hat{x})^T\omega \geq 0$ for all $\omega \in T_\Omega(\hat{x})$, and so $-\nabla f(\hat{x}) \in N_\Omega(\hat{x})$. Thus, $\hat{x}$ is a Karush-Kuhn-Tucker point.*

Proof. Lemma 2.3 and the strict differentiability of f at $\hat{x}$ ensure that $\nabla f(\hat{x})^T d \geq 0$ for all $d \in D \cap T_\Omega(\hat{x})$. Since D includes all the tangent cone generators, and each is used infinitely often in K, it follows that every $\omega \in T_\Omega(\hat{x})$ can be represented as a nonnegative linear combination of $D \cap T_\Omega(\hat{x})$; thus, $\nabla f(\hat{x})^T\omega \geq 0$ for all $\omega \in T_\Omega(\hat{x})$. To complete the proof, we multiply both sides by -1 and conclude that $-\nabla f(\hat{x}) \in N_\Omega(\hat{x})$. ∎

4.3.2. Generating directions. While the new hypothesis of Theorem 4.6 is weaker and makes the result more general than Theorem 2.4, it is more difficult to enforce. The enumeration scheme mentioned above will ensure that the tangent cone generators get used infinitely often, but it cannot ensure that they get used infinitely often *in the refining subsequence*. Thus we make the following additional assumption.

A4: Any direction used infinitely often is also used infinitely often in any refining subsequence.

This assumption is actually quite mild, since a direction that does not satisfy A4 would always be successful (infinitely often) after a finite number of iterations.

The following result establishes an important connection between the constraints and tangent cone generators.

LEMMA 4.7. *Let $\hat{x}$ be the limit of a subsequence of GPS iterates, and let $\hat{S}$ and $\hat{D}$ be the sets of active constraints and tangent cone generators, respectively, at $\hat{x}$. If every constraint in $\hat{S}$ is used to form tangent cone generators infinitely often in the subsequence, then every tangent cone generator in $\hat{D}$ is also used infinitely often in the same subsequence.*

Proof. Let $\hat{S}_j$, $j = 1, \ldots, s$ be maximally linearly independent subsets of $\hat{S}$, such that $\hat{S} = \bigcup_{j=1}^{s} \hat{S}_j$. Furthermore, let $D(\hat{S}_j)$ denote the set of tangent cone generators produced by only the constraints in $\hat{S}_j$. Price and Coope [21] show that $\hat{D} \subset \bigcup_{j=1}^{s} D(\hat{S}_j)$. Thus, if $\hat{S}_j$ is used infinitely often, then $D(\hat{S}_j)$ is used infinitely often, and if every $\hat{S}_j$, $j = 1, \ldots, s$, is used infinitely often, then every direction in $\hat{D}$ is used infinitely often. ∎

We now give two examples of approaches that generate directions satisfying the hypotheses of Theorem 4.6, followed by convergence theorems for each.

Random Selection: Randomly select (with uniform probability) r linearly independent ε-active constraints to form tangent cone generators.

Sequential Selection: Order the s subsets of r linearly independent elements of S_k as $S_i, i = 1, \ldots, s$, and use subset S_j, $j = 1 + k \mod s$, at iteration k.

THEOREM 4.8. *Let $\hat{x}$ be the limit of a refining subsequence $\{x_k\}_{k\in K}$, in which the set of nonredundant binding constraints at $\hat{x}$ is linearly dependent. If search directions are obtained by Random Selection whenever the elements of S_k are linearly dependent, then with probability 1, all tangent cone generators at $\hat{x}$ will be used infinitely often in K.*

Proof. For any nonredundant active constraint at $\hat{x}$, let P_k denote the probability that the constraint is randomly selected at iteration k. Then for sufficiently large k, the set S_k is fixed with p elements (corresponding to the active constraints at $\hat{x}$), and $P_k = \frac{r}{p}$. Then the probability that the constraint is selected infinitely often in any infinite subsequence M of iterates (with sufficiently large k) is equal to $1 - \prod_{k\in M}(1 - P_k) = 1 - \prod_{k\in M}\frac{p-r}{p} = 1$. The result then follows from Lemma 4.7. □

THEOREM 4.9. *Let $\hat{x}$ be the limit of a refining subsequence $\{x_k\}_{k\in K}$, in which the set of nonredundant binding constraints at $\hat{x}$ is linearly dependent. Under assumption A4, if search directions are obtained by Sequential Selection whenever the elements of S_k are linearly dependent, then all tangent cone generators at $\hat{x}$ will be used infinitely often in K.*

Proof. Since subset S_j, $j = 1 + k \mod s$ is used at iteration k, S_j is used infinitely often in the iteration sequence. Furthermore $S_j \subset \hat{S}$ for all sufficiently large k, where $\hat{S}$ is the set of active constraints at $\hat{x}$. The result follows from Assumption A4 and Lemma 4.7. □

In each of these two instances, something is sacrificed for the sake of implementation - either a weaker convergence result (Random Selection) or the additional assumption A4 (Sequential Selection). However, if function evaluations are expensive, the alternative of identifying all the tangent cone generators at each iteration will become intractable.

Furthermore, this by no means exhausts the possibilities for choosing tangent cone generators when nonredundant constraints are linearly dependent. Considering that the projection method measures distance to each constraint boundary, one promising alternative is to select the closest $n - 1$ constraints (with ties broken arbitrarily), plus one more constraint obtained by either random or sequential selection. The latter constraint allows the theory in the previous two theorems to hold, while offering an intelligent heuristic in selecting those constraints that are closer to the current iterate. Choosing the closest constraints is equivalent to reducing ε at each iteration so that fewer constraints are flagged as ε-active.

4.4. An algorithm for constructing the set of generators. In this section, we present an algorithm for constructing a set of generators for the cone $K^\circ(x_k, \varepsilon)$ at the current iterate x_k for a given parameter ε.

4.4.1. Comments on the algorithm. The algorithm consists of two main parts. In the first part, we determine the set of indices of the nonredundant ε-active constraints $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$ and form the matrix B_N whose columns are the columns of A corresponding to the indices in $I_N(x_k, \varepsilon)$. We use information about the set $I_N(x_k, \varepsilon)$ from the previous iterations of the GPS algorithm. Namely, we put into the set $I_N(x_k, \varepsilon)$ all indices that correspond to the ε–active constraints at the current iterate and that were detected as indices of the nonredundant constraints at the previous iterations of the algorithm. In the second part of the algorithm, we construct the set of generators D_k required by GPS and by Theorem 2.4.

First, we try to identify the nonredundant active constraints. If the matrix B_k defined by (4.1) has full rank, then all ε-active constraints are nonredundant, $I_N(x_k, \varepsilon) = I(x_k, \varepsilon)$, and $B_N = B_k$. If the matrix B_k does not have full rank and we have indices that have not been classified at the previous iterations of the algorithm, we propose using two steps in succession.

The first strategy is intended to determine nonredundant constraints cheaply by applying the projection method described in section 3.3.1. By Proposition 3.16, if the projection $P_j(x_k)$ of the current iterate x_k onto the hyperplane $H_j = \{x \in \mathbb{R}^n : \bar{a}_j^T x = \bar{b}_j\}$ is feasible, and only the jth constraint holds with equality at $P_j(x_k)$, then the jth constraint is nonredundant, and we can put index j into the set $I_N(x_k, \varepsilon)$. If some constraints have not been identified by the projection method, we can either apply the projection method with some other point $\tilde{x} \neq x_k$ or apply the second strategy.

The second strategy is intended to classify redundant and nonredundant constraints among those that have not already been determined as nonredundant by the projection method. To identify each constraint, the approach outlined in [10] and in Section 3.15 is applied. If the number of constraints to be identified is too large, we can skip an application of this strategy and construct a set of generators using the set $I_N(x_k, \varepsilon)$ obtained from the first strategy. Then, while performing the poll step, if we find some point $\bar{x} = x_k + \Delta\bar{d}$, where $\bar{d}$ is some column of D_k, such that $a_j^T \bar{x} > b_j$ and $a_i^T \bar{x} \leq b_i$ for all $i \in I(x_k, \varepsilon)\backslash\{j\}$, we can conclude that $\Omega(x_k, \varepsilon) \subsetneq \Omega_j(x_k, \varepsilon)$. Hence, by Corollary 3.13, the jth constraint is nonredundant, and we add j to the set $I_N(x_k, \varepsilon)$.

Once we have specified all redundant and nonredundant constraints, we compose the matrix B_N of those columns of A that correspond to nonredundant constraints. The rank of B_N can be determined by QR factorization. If B_N has full rank, then we construct the set of generators using QR or LU factorization. If B_N does not have full rank, we construct the set of generators from a set of linearly independent columns of B_N, and as the iteration sequence progresses, we invoke one of the methods described in Section 4.3 to ensure that all maximally linearly independent subsets get used infinitely often.

4.4.2. Algorithm. We denote the set of indices of the nonredundant ε-active constraints by $I_N(x_k, \varepsilon)$. Thus, for $j \in I(x_k, \varepsilon)$,

1. if $j \in I_N(x_k, \varepsilon)$, then the inequality $a_j^T x \leq b_j$ is nonredundant; and
2. if $j \in I(x_k, \varepsilon)\backslash I_N(x_k, \varepsilon)$, then the inequality $a_j^T x \leq b_j$ is redundant.

We use $I_N \subseteq I$ to denote the set of indices that are detected as nonredundant at some iteration of the algorithm. Thus $I_N = \emptyset$ in the beginning of the algorithm.

We denote the rational matrix in (1.2) by A^T and the scaled matrix defined in (3.5) by $\bar{A}^T$. The matrix B_k is defined by (4.1) and is composed of columns a_j of A, where $j \in I(x_k, \varepsilon)$, while the matrix B_N is composed of those columns of A whose indices are in the set $I_N(x_k, \varepsilon)$. Thus, the columns of B_N are those vectors normal to the nonredundant constraints.

Algorithm for constructing the set of generators D_k.

Let the current iterate $x_k \in \mathbb{R}^n$ and a parameter $\varepsilon > 0$ be given.

% *Part I: Constructing the set* $I_N(x_k, \varepsilon)$

% *Construct the working index set* $I(x_k, \varepsilon)$

for $i = 1$ **to** $|I|$
  if $0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon$
    $I(x_k, \varepsilon) \leftarrow I(x_k, \varepsilon) \cup \{i\}$
    $B_k \leftarrow [B_k,\ a_i]$
  endif
endfor

if $\text{rank}(B_k) = |I(x_k, \varepsilon)|$ % *the case where all constraints are nonredundant*
  $I_N(x_k, \varepsilon) \leftarrow I(x_k, \varepsilon)$
  $B_N \leftarrow B_k$

else

% *using information from the previous iterations of the algorithm*
for each $j \in \{I(x_k, \varepsilon) \bigcap I_N\}$
$I_N(x_k, \varepsilon) \leftarrow I_N(x_k, \varepsilon) \cup \{j\}$
$B_N \leftarrow [B_N,\ a_j]$
endfor

% *Identification of the nonredundant and redundant constraints*
for each j **in** $\{I(x_k, \varepsilon) \backslash I_N(x_k, \varepsilon)\}$

% *the first strategy*
$P_j(x_k) = x_k + \bar{a}_j(\bar{b}_j - \bar{a}_j^T x_k)$ % *see Lemma 3.4*

if $\bar{a}_i^T P_j(x_k) < \bar{b}_i$ **for all** $i \in I \backslash \{j\}$
$I_N(x_k, \varepsilon) \leftarrow I_N(x_k, \varepsilon) \cup \{j\}$
$B_N \leftarrow [B_N,\ a_j]$
$I_N \leftarrow I_N \cup \{j\}$

else

% *the second strategy*
solve LP problem (3.15) for x^*

if $a_j^T x^* \leq b_j$ % *the jth constraint is redundant*
remove $a_j x \leq b_j$ from Ω
$I \leftarrow I \setminus \{j\}$
$I(x_k, \varepsilon) \leftarrow I(x_k, \varepsilon) \setminus \{j\}$
else % *the jth constraint is nonredundant*
$I_N(x_k, \varepsilon) \leftarrow I_N(x_k, \varepsilon) \cup \{j\}$
$B_N \leftarrow [B_N,\ a_j]$
$I_N \leftarrow I_N \cup \{j\}$
endif
endif
endfor

endif

% *Part II: Constructing the set of generators* D_k

$r = \text{rank}(B_N)$

if $r \neq |I_N(x_k, \varepsilon)|$ % *degenerate case*
$B_N \leftarrow B$, where B is composed of r linearly independent columns of B_N

endif
$V = B_N$
$D_1 \leftarrow V(V^T V)^{-1}$
$D_2 \leftarrow I - V(V^T V)^{-1} V^T$
$D = [D_1,\ D_2,\ -D_1,\ -D_2]$

As discussed in Section 4.2, the construction of the directions in D, in practice, can be done making use of either LU decomposition, as suggested by Lewis and Torczon [16], or by the more efficient QR factorization approach presented in Section 4.2. In the latter case, D_1 and D_2 are computed according to Corollary 4.5.

We should point out that, in practice, the choice of ε can have a significant affect on numerical performance. If the value is set too low, then the mesh size may become very small before appropriate conforming directions are generated. If this happens, the algorithm may then progress along a new conforming direction,

but with the significantly reduced mesh size, resulting in a lot more function evaluations. On the other hand, too large a value may mark too many constraints as active. This could result in otherwise good directions being replaced by worse ones, and even a false detection of degeneracy, resulting in additional unnecessary function evaluations.

4.4.3. Numerical Tests. To test the algorithm, we formed five test problems with varying numbers of variables and redundant linear constraints to test the ability of our approach to accurately construct the set $I_N(x_k, \varepsilon)$ of nonredundant constraints. In doing so, we chose a trial point x_k close to several of the constraints and tested the ability of our algorithm to identify the nonredundant ones. The test problems are described as follows:

Test Problem 1: Same as the problem given in (4.2), with a test value of $x_k = (0.1, 0.1, 0.1)^T$.
Test problem 2: The following problem with a test value of $x_k = (0.01, -0.01, -0.01, -0.00001, 0.01)^T$:

$$\begin{aligned} -x_1 + x_2 &\leq 0 \\ x_1 + x_2 &\leq 1 \\ x_2 + x_3 + x_4 &\leq 0 \\ -x_2 + x_5 &\leq 5 \\ -x_1 + x_2 &\leq 0 \\ x_3 &\leq 0 \\ -0.8x_1 + x_2 + x_3 &\leq 0. \end{aligned}$$

Test Problem 3: The following problem with a test value of $x_k = (0.01, 0.01, 0.01, 0.01, 0.01)^T$:

$$\begin{aligned} x_1 - 2x_2 - 2x_3 + x_4 &\leq 0 \\ -2x_1 + x_2 - 2x_3 &\leq 0 \\ -2x_1 - 2x_2 + x_3 &\leq 0 \\ -x_1 &\leq 0 \\ -x_2 &\leq 0 \\ -x_3 - x_5 &\leq 0 \\ -x_4 - 0.1x_5 &\leq 0. \end{aligned}$$

Test Problem 4: Same as Test Problem 3, but with $x_k = (0.000001, 0.000001, 0.000001, 0.000001, 0.1)^T$.
Test Problem 5: Same as Test Problem 3, but with $x_k = (0.001, 0.001, 0.001, 0.001, 0.001)^T$.

We report results in Table 4.1, where each row corresponds to one of the five test problems (in the order presented), and where the number of variables is given in the first column. Columns 2 and 3 show the number of nonredundant and redundant constraints, respectively, with their sum representing the total number of constraints for each problem. The last two columns indicate how many of the constraints were identified as nonredundant, first by the projection method, and then by the LP approach if projection failed to identify all the nonredundant ones. As is shown in the table, the projection method identifies most of the nonredundant constraints, and with the LP method as a backup, all the constraints are correctly identified.

With this approach in place, the number of GPS iterations required for a problem with no redundant constraints will be no different than for a modified version of the same problem, in which any number of additional redundant constraints are added, since the algorithm detects and removes the redundant constraints at each iteration.

Finally, we coded up the random selection and sequential selection approaches for handling linearly dependent, nonredundant ε-active constraints (see Section 4.3), added this code to the NOMADm software [1]), and tested the approaches on the following test problem:

Test problem 6:

$$\min_x \ -x_1^2 - x_2^2 - x_3^2$$

TABLE 4.1
Constructing the set $I_N(x_k, \varepsilon)$ at the current iterate x_k

	Constraints		Detected as nonredundant	
Variables	Nonredundant	Redundant	by Projection	by LP approach
3	6	0	6	
5	6	1	5	1
5	7	0	6	1
5	7	0	5	2
5	7	0	6	1

$$
\begin{aligned}
&\text{subject to}\\
4x_1 + 4x_2 - 3x_3 &\geq 0\\
16x_1 + 8x_2 - 9x_3 &\geq 0\\
24x_1 + 8x_2 - 11x_3 &\geq 0\\
8x_1 - 24x_2 + 23x_3 &\geq 0\\
8x_1 + 8x_2 - 13x_3 &\leq 0\\
24x_1 + 8x_2 - 25x_3 &\leq 0\\
24x_1 - 8x_2 - 21x_3 &\leq 0\\
x_1 + 2x_2 - x_3 &\geq 0\\
x_1 + x_2 + x_3 &\leq 8.
\end{aligned}
$$

Test Problem 6 has a degenerate global maximizer at the origin, which is *not* the local solution. We start the algorithm there, and in order to avoid stalling there, a set of $n = 3$ constraints must be chosen that will generate a feasible descent direction. Since the algorithm does not evaluate the objective at infeasible points, and the cones of feasible directions and of descent directions coincide at this point, moving off the degenerate point will always occur at the second function evaluation. Thus our measure of performance becomes the number of iterations, rather than function evaluations, required to move off the degenerate point. The number of iterations gives a measure of how many unsuccessful attempts the algorithm made at including a feasible descent direction in its set of poll directions.

For sequential selection, we paired the constraints in consecutive order; *i.e.*, constraints 1 and 2, 1 and 3, 1 and 4, etc. The algorithm required 9 iterations to move off of the degenerate point. For random selection, we performed 10 replications and achieved the following number of iterations to move off the degenerate point: $\{4, 3, 4, 2, 2, 2, 5, 7, 2, 6\}$.

We tested both approaches with default settings of $\Delta_0 = 1$, a mesh refinement strategy of $\Delta_{k+1} = \frac{1}{2}\Delta_k$, and empty SEARCH step. The variable ε was set to 10^{-4} (although this had no bearing on performance, since we started at the degenerate point), and the QR factorization was used in constructing tangent cone generators. Both direction selection approaches successfully moved off the degenerate point. A comparison between the two approaches is not particularly relevant, since the result for sequential selection is highly dependent on the order in which constraints are expressed or chosen. In the worst-case, since each iteration required $2n = 6$ directions, consideration of between 12 and 54 directions were required to move off the degenerate point. However, had we attempted to compute all the directions during the first iteration, we might have had to consider, by (4.7), a worst-case 84 subsets of constraints, or 504 directions.

As an aside, we also let the algorithm run to completion (termination tolerance of $\Delta_k < 10^{-8}$), and all 11 runs successfully converged to one of two local minimizers (at approximately $(3.67, 0.221, 4.11)^T$ and $(0.471, 3.76, 3.76)^T$). Sequential selection required a total of 67 iterations and 78 function evaluations, while random selection required 55–91 iterations and 84–144 function evaluations.

We should point out that, for this problem, there is no real cost savings in terms of function evaluations because the infeasible points were not evaluated. Had we chosen a problem in which the initial point was degenerate, but neither a minimizer or maximizer, then the number of function evaluations would become a factor. In this case, the selection of certain combinations of constraints would generate feasible non-descent directions, which would result in additional function evaluations at points with worse function values. But even without the cost savings, we still avoid the potentially intractable task of specifically identifying all the tangent cone generators at each iteration via some vertex enumeration scheme.

5. Nonlinearly constrained minimization. The goal of this section is to illustrate how the projection approach proposed in this paper can also be effective for handling degeneracy in nonlinearly constrained optimization problems. In doing so, we should point out that our approach is different than that of [20] and [26] (and others cited in these papers). Both approaches use local information about the (twice continuously differentiable) objective and constraint functions to identify active constraints. Moreover, the focus in [26] is on distinguishing between strongly and weakly active constraints – the latter having Lagrange multiplier values of zero. In our case, we do not have multiplier values available, and even if we did, most direct search methods we might consider using, such as [12] and [17], can handle weakly active constraints transparently if constraint gradients are linearly independent.

We consider the nonlinearly constrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad x \in \Omega = \{x \in \mathbb{R}^n : c_i(x) \le 0,\ i = 1, \ldots, q\}. \tag{5.1}$$

All constraint functions $c_i, i = 1, 2, \ldots, q$, are assumed to be continuously differentiable, but their gradients may not be available. The algorithm in [17] uses constraint gradients, while the one in [12] uses only approximations. Our intent is to be as general as possible, so that the ideas presented here might be extendable to both algorithms, as well as other direct search methods.

Similar to Section 3, the region defined by all but the j-th constraint is given by

$$\Omega_j = \{x \in \mathbb{R}^n : c_i(x) \le 0, i \in I\backslash\{j\}\},$$

where $I = \{1, 2, \ldots, q\}$. Additionally, for $\delta > 0$, we define $U_\delta(x) = \{y \in \mathbb{R}^n : \|y - x\| \le \delta\}$, and offer a definition of local redundancy (nonredundancy), in the sense that the constraints are locally nonredundant if they define the shape of the feasible region in some neighborhood of a point $x \in \mathbb{R}^n$. This is illustrated in Figure 5.1.

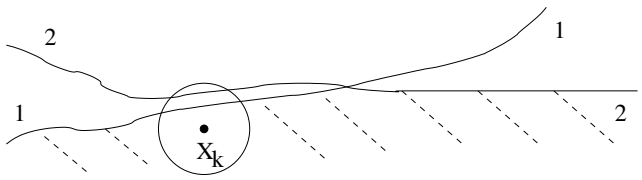

FIG. 5.1. *An illustration of a locally redundant constraint. Constraint* 2 *is locally redundant at* x_k.

DEFINITION 5.1 (Locally redundant constraint). *The* j*th constraint* $c_j(x) \le 0$ *is* locally redundant *at* x *in the description of* Ω *if, for some* $\delta > 0$, $\Omega \cap U_\delta(x) = \Omega_j \cap U_\delta(x)$, *and is* locally nonredundant *otherwise.*

Our main interest is in the problem of constructing search directions that conform to the boundary of Ω. First, we define constraint j as ε-*active* if $-\varepsilon \le c_j(x) \le 0$. For the iterate x_k at iteration k, we denote by $I(x_k, \varepsilon)$ the set of indices of the ε-active constraints at x_k; namely,

$$I(x_k, \varepsilon) = \{j = 1, 2, \ldots, q : -\varepsilon \le c_j(x_k) \le 0\},$$

and extend the following from similar definitions given in Section 3:

$$\Omega(x_k, \varepsilon) = \{x \in \mathbb{R}^n : c_i(x) \leq 0,\ i \in I(x_k, \varepsilon)\},$$
$$\Omega_j(x_k, \varepsilon) = \{x \in \mathbb{R}^n : c_i(x) \leq 0,\ i \in I(x_k, \varepsilon)\backslash\{j\}\}, \quad j \in I(x_k, \varepsilon).$$

If x_k is close to the boundary of Ω, then the set of directions should contain generators for the tangent cone $T_\Omega(x_k)$ for boundary points near x_k.

We assume that estimates $a_i^{(k)}$ of the gradients $\nabla c_i(x_k)$, $i = 1, 2, \ldots, q$, are available. Thus, an estimate $C^{(k)}(I(x_k, \varepsilon))$ of the tangent cone $T_\Omega(x_k)$ is given by

$$C^{(k)}(I(x_k, \varepsilon)) = \{v \in \mathbb{R}^n : v^T a_i^{(k)} \leq 0 \quad \forall i \in I(x_k, \varepsilon)\}. \tag{5.2}$$

By Definition 4.1, each of these cones can be expressed as the set of nonnegative linear combinations of a finite number of generators, $\{v_j\}_{j=1}^p \subset \mathbb{R}^n$.

One of the main assumptions in [12] is that at each point x on the boundary of Ω, the gradients of the constraints active at x are linearly independent. By extending our ideas from previous subsections to the nonlinear case, this assumption can be relaxed.

The next proposition is simply an application of Proposition 4.2 to the cone defined by the linearized constraints, except that the index sets apply to nonlinear constraints. As a consequence, it is sufficient to construct the set of generators for only the locally nonredundant constraints.

PROPOSITION 5.2. *Let $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$ be the subset of indices of the locally nonredundant constraints that define $\Omega(x_k, \varepsilon)$. Let the cone $C^{(k)}(I(x_k, \varepsilon))$ be defined by* (5.2) *and let the cone $C_N^{(k)}$ be given by $C_N^{(k)} = \{v \in \mathbb{R}^n : v^T a_i^{(k)} \leq 0 \quad \forall i \in I_N(x_k, \varepsilon)\}$. If $\{v_1, \ldots, v_p\}$ is a set of generators for $C_N^{(k)}$, then it is also a set of generators for $C^{(k)}(I(x_k, \varepsilon))$.*

Proof. This follows directly from Corollary 3.13. □

To extend the projection approach described in Section 3.3.1 for detecting locally nonredundant nonlinear constraints, we simply project onto a linearization of the constraint boundary, based on approximations to the constraint gradients at the current iterate; *i.e.*, we project onto the hyperplane $H_j = \{v \in \mathbb{R}^n : v^T a_j^{(k)} = 0\}$. Scaling the constraints similar to (3.5) and applying Lemma 3.4 yields a projection equation similar to (3.10); namely,

$$P_j(x_k) = x_k + \bar{a}_j^{(k)} c_j(x_k), \quad \bar{a}_j^{(k)} = \frac{a_j^{(k)}}{\|a_j^{(k)}\|}, \quad j = 1, 2, \ldots, q. \tag{5.3}$$

If the generators of $C_N^{(k)}$ at iteration k are linearly independent, then they would all be included in the set of search directions for that iteration. Otherwise, the set of search directions would include a maximal linearly independent subset of the generators, selected in exactly the same manner as discussed in Section 4.3.

We omit a formal discussion of convergence, since any results would be dependent on the algorithm being used and on the details of its implementation. However, it appears safe to assume that any convergence results will require a certain degree of accuracy by the vectors $a_j^{(k)}$ as approximations to the constraint gradients $\nabla c_j(x_k)$.

We view these ideas as a natural extension of those of Section 3.3.1, and one that can achieve a significant cost savings over the LP approach. Recall from Section 3.3.2 that the expense of the LP approach for linear constrained problems can be circumvented by performing it before the algorithm commences, since the redundancy of each constraint is independent of the location of the current iterate. However, this is not true for nonlinear constraints, in which case, the LP approach would have to be performed at every iteration, which is considerably more expensive than projection.

6. Concluding remarks. This paper fills an important gap in the pattern search literature, complementing the previous work of Lewis and Torczon [16] by rigorously treating the case of degenerate linear constraints. We have introduced an inexpensive projection method for identifying nonredundant constraints, which, when used in conjunction with a linear programming approach as a backup, can cheaply assess the redundancy of each constraint, and thus aid pattern search in computing directions that conform to the boundary of the feasible region. For the case in which nonredundant ε-active constraints are linearly dependent, we avoid complete enumeration of tangent cone generators by including only a subset of them, and then changing them at each iteration, such that all are used infinitely often over the entire iteration sequence. We prove first-order convergence in this case, under an additional mild assumption. Finally, we have shown how our ideas can be extended to nonlinearly constrained optimization problems under similar degenerate conditions.

Acknowledgements. This work was begun at the IMA, where Olga Brezhneva was a postdoctoral fellow, John Dennis was a long-term visitor, and Mark Abramson was a short-term visitor. We thank the IMA for providing such a fine atmosphere for collaboration. We also thank Charles Audet for many useful discussions and suggestions, and two anonymous referees for comments that have helped us improve the paper.

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, United States Government, or research sponsors.

REFERENCES

[1] M. Abramson, NOMADm Optimization Software, `http://en.afit.edu/ENC/Faculty/MAbramson/NOMADm.html`, 2003.
[2] M. Abramson, *Second-order behavior of pattern search*, SIAM J. Optim., *to appear*. Also appears as Technical Report TR04-03, Rice University, Department of Computational Mathematics, 2004.
[3] C. Audet, *Convergence results for pattern search algorithms are tight*, Optim. Engin., 5 (2004), pp. 101–122.
[4] C. Audet and J. E. Dennis JR., *Analysis of generalized pattern searches*, SIAM J. Optim., 13 (2003), pp. 889–903.
[5] D. M. Avis and K. Fukuda, *A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra*, Discrete Comput. Geom., 8 (1992), pp. 295–313.
[6] D. M. Avis and K. Fukuda, *Reverse search for enumeration*, Discrete Applied Mathematics, 6 (1996), pp. 21-46.
[7] H. C. P. Berbee, C. G. E. Boender, A. H. G. R. Kan, C. L. Scheffer, R. L. Smith, and J. Telgen, *Hit-and-run algorithms for the identification of nonredundant linear inequalities*, Math. Program., 37 (1987), pp. 184–207.
[8] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, Belmont, MA, 1999.
[9] A. Boneh, S. Boneh, and R. J. Caron, *Constraint classification in mathematical programming*, Math. Program., 61 (1993), pp. 61–73.
[10] R. J. Caron, J. F. McDonald, and C. M. Ponic, *A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary*, J. Optim. Theory Appl., 62 (1989), pp. 225–237.
[11] F. H. Clarke, *Optimization and nonsmooth analysis*, SIAM Classics in applied mathematics Vol.5 (1990), Philadephia.
[12] I. D. Coope, J. E. Dennis Jr., and C. J. Price, *Direct search methods for nonlinearly constrained optimization using filters and frames*, Optim. Engin., 5 (2004), pp. 123–144.
[13] M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, *Redundancy in mathematical programming*, Springer-Verlag, Berlin, 1983.
[14] T. K. Kolda, R. M. Lewis and V. Torczon, *Optimization by direct search: new perspectives on some classical and modern methods*, SIAM Review, 45 (2003), pp. 385–482.
[15] T. K. Kolda, R. M. Lewis and V. Torczon, *Stationarity results for generating set search for linearly constrained optimization*, Technical Report SAND2003-8550, Sandia National Laboratories, Livermore, California, Oct. 2003.
[16] R. M. Lewis and V. Torczon, *Pattern search methods for linearly constrained minimization*, SIAM J. Optim., 10 (2000), pp. 917–941.
[17] S. Lucidi, M. Sciandrone, and P. Tseng, *Objective-Derivative-Free Methods for Constrained Optimization*, Math. Program., 92 (2002), pp. 37–59.
[18] J. H. May, *Linearly constrained nonlinear programming: a solution method that does not require analytic derivatives*, PhD thesis, Yale University, December 1974.
[19] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, John Wiley & Sons, New York, 1988.
[20] C. Oberlin and S. J. Wright, *Active constraint identification in nonlinear Programming*, Optimization Technical Report 05-01, Computer Sciences Department, University of Wisconsin-Madison, January, 2005.

[21] C. J. Price and I. D. Coope, *Frames and grids in unconstrained and linearly constrained optimization: a non-smooth approach*, SIAM J. Optim., 14 (2004), pp. 415–438.
[22] V. Torczon, *On the convergence of pattern search*, SIAM J. Optim., 7 (1997), pp. 1–25.
[23] L. N. Trefethen and D. Bau, III, *Numerical linear algebra*, SIAM, Philadelphia, 1997.
[24] J. Van Tiel, *Convex Analysis*, John Wiley & Sons, New York, 1984.
[25] L. A. Wolsey, *Integer programming*, John Wiley & Sons, New York, 1998.
[26] S. J. Wright, *Constraint identification and algorithm stabilization for degenerate nonlinear programs*, Math. Program., Series B, 95 (2003), pp. 137-160.